This document includes the contents of the two email messages that I sent to students via email before the end of the Winter term.

CSSE 230 Spring, 2008. Information for students.

You may want to act on some of this before you leave for break.

If you have questions or concerns about any of this information, feel free to reply to this email, call me ad x8331, or come to my office, F-210.

You have probably heard that the CSSE department is revamping our Software Fundamentals course sequence in 2007-08, starting with 120 in the Fall, 220 this term, and 230 in the Spring. Last Fall we sent out an advisory recommending that students who took 220 or 221 prior to this academic year should take 230 in the Winter if at all possible, since the prerequisite material will change for the Spring 230. I realize that some of you were unable to fit the course into your schedule then.

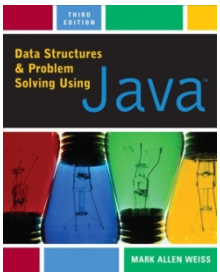
The remainder of this message is intended primarily for students who took the "old" 220 or 221 (prior to Fall, 2007), but I thought perhaps everyone in the class might benefit from reading it. I want to tell you about the contents of the current CSSE 220, which will be the prerequisite material for next term's 230. I will allow four days at the beginning of the course before I expect you to be caught up on it, but during that time, there will be a normal (normal for *my* classes, and you have probably heard the rumors about what that means!) collection of new work for you to do as well, so I recommend that you consider looking at most of the remedial material before the term starts.

I want to help you now to be able to make the transition from the old version of 220 to the new version of 230. I do not want you to be at a disadvantage when you come into 230. That is the reason for this message. Because of the change in the 220 course material, you will have to do some extra work (how much work that is depends on when you took 220 and how well you understood and still retain the material and skills from that course). There is no way around that. You will be much better off if you do a lot of that work before the term begins.

The #1 thing you need to bring into 230 is being ready to jump in and write Java programs using the Eclipse environment. If it has been a few months since you did that, you should play with the environment a little before the term starts so you are familiar with it on Day 1. If you were very confident about programming in Java at the end of 220, it should come back to you quickly. If not, I suggest that you write a couple hundred lines of java code before March 3. A major goal of 220 for me is that students are competent, confident, independent Java programmers by the end of the course. If you are not there yet, I'll be happy to help you, but you should try to get a head start before the term begins. Some suggested programs to write appear later in this document.

You need up-to-date **software** (and familiarity with how to use it). This includes Java 6 (Students who took 221 or 220 during 2006-07 year are very likely to have an earlier version), Eclipse 3.2 (if you haven't used the Eclipse debugger in a while, be sure to get some practice with it), Subclipse, and Violet. Installation instructions for all of these are at http://www.rose-hulman.edu/class/csse/resources/ . I recommend that you install this software and check it out to make sure it works for you so you can get help if needed before you leave campus for break.
In addition, you should know how to use a program (such as AFS Drive Mapper or Network Identity Manager or SecureFX) to place files on AFS, and how to log into addiator.rose-hulman.edu using SecureCRT, puTTY, or some other ssh program.
You should know (and you should find out if you don't already know) how to write Java programs that expect and use command-line arguments, and how to run those programs form Eclipse and from the DOS command-line.

Textbook It's the same book we used for 220 in the Fall and Winter terms and 221 in the Fall. Mark A Weiss, Data Structures and Problem Solving with Java, 3rd Edition.



You should get it from the bookstore (or order it online) before you leave campus. You need to look through this book so you can figure out which topics from the current 220 you need to review or learn. If there are a lot of them, you will need to do some of that during the break. Plan to read those all of those topics carefully between now and March 7. **The most important parts of the book for you to know in order to be prepared for 230 are chapters 4 and 6, plus sections 2.4, 5.1, 5.2, and 5.6.**

220 textbook coverage and other major topics Here is the list of 220 material that I gave my students in preparation for the Final exam.

- Weiss chapters 1-4 (except 4.7.5, 4.7.6)
- Sections 5.1, 5.2, 5.4-5.8 (you do not have to know the formal definition of big-oh and its cousins or do any proofs, but you should know the informal definitions)
- Chapter 6 (this is the most important chapter to know in preparation for 230. You should know about all of these data structures, how to use them, how to choose which to use, but not necessarily how to implement them.
- Sections 7.1, 7.3, 8.1-8.5 (not 8.4.1)
- GUIs and Events (partly covered in the Weiss appendices, but we did more with them).
- Implementation of Stack, Queue, ArrayList, and especially LinkedList, but not all of the details I Chapters 15-17)
- Binary Files, Random access, Object I/O, Networking
- Unit tests using JUnit: writing them and running them. Why do we do unit tests?

Within the next several days, you should quickly skim these sections of this book, making a list of the things you do not yet know or that you will need to review before or soon after the course begins. Don't fool yourself, thinking you will have lots of free time that first week of classes. You know it never works out that way!
Even some of the "intro to Java" chapters (1-4) may have quite a few things that you do not know if you have not already read this book. For example, do you know about static initializers, the "static import directive", StringTokenizer, the decorator pattern, wrapper classes, generic static methods, wildcards, function objects, Comparators, nested and local classes, dynamic dispatch, and BufferedReader?
You should know the general idea of what big-Oh, big-Omega, and big-Theta are about, and in particular, what are the relationships among these three concepts. You should know how to get the big-Theta running time for simple code snippets. You do not have to know the formal definition of big-oh. This material is in Chapter 5.
You should know about the Java Collections framework, including the Comparable, Comparator, Collection, Iterator, List, ListIterator, Set, and Map interfaces, as well as the ArrayList, LinkedList, TreeSet, HashSet, TreeMap, HashMap, Collections, and Arrays classes. These are in Chapter 6.
You should be beginning to feel comfortable with writing recursive methods. (sections 7.1 and 7.3)
You should know five elementary sorting algorithms: Insertion, Selection, Bubble, Merge, and Shell. (See chapter 8)

Written problems played a role in 220, and they will play a bigger role in 230. For your convenience, I made a single document that contains all of the assigned written problems from Winter 220. You should look at all of those problems. If there are any that you are not sure you can easily do, you should try them. Most are straightforward, but a few require some creative thought. http://www.rose-hulman.edu/class/csse/csse220/200820/web/Homework/All-written.html

Programming problems. You should take a look at a few of the programming problems that I assigned. Once again, if you are not sure that you can do them, you should actually do them.

Dots http://www.rose-hulman.edu/class/csse/csse220/200820/web/Programs/Dots/ A simple GUI program that responds to events.
Hardy's Taxi: http://www.rose-hulman.edu/class/csse/csse220/200820/web/Programs/HardysTaxi/HardysTaxiAssignment.doc This is a short program that requires a bit of thought. Can you make it fast enough to do N=500? N=100? N=2000? Can you be sure that you really get the Nth smallest number?
LinkedList: Create a class that implements most of the methods of the **java.util.List** interface (which extends **java.util.Collection**), using a linked list with a header node.
Markov. http://www.rose-hulman.edu/class/csse/csse220/200820/web/Programs/Markov/index-Markov.html This one was all about understanding and using the classes that are part of the Java Collections Framework. This assignment used to be part of 230, but we moved it to 220. It is an excellent (and challenging) exercise. I highly recommend that you do it (perhaps you can find another "old 220" student to do it with you).

Terminology. There is a fairly extensive vocabulary associated with Object-oriented programming and Java. We "speak that language" a lot in class, and so it is important for you to know the terminology. In 220, we had a "Key Concepts Quiz", to check students knowledge of the terminology from the Key Concepts sections that appear at the ends of Weiss Chapters 1-4. You should also know the Key Concepts from chapter 6.

Useful materials from this term's 220 class

Syllabus. http://www.rose-hulman.edu/class/csse/csse220/200820/web/syllabus.html In particular, this has a number of links to Java resources (in particular to Safari Books On-Line). It will also be a model for my 230 syllabus.
PowerPoint slides from class. http://www.rose-hulman.edu/class/csse/csse220/200820/web/Slides/ These will give you an idea of the level at which we discussed many course topics.
Schedule page: http://www.rose-hulman.edu/class/csse/csse220/200820/web/Schedule/Schedule.htm includes links to many other things
Resources: http://www.rose-hulman.edu/class/csse/csse220/200820/web/Resources/

Excerpt from a note I sent to students two weeks before the last time I taught 230:

Weiss Chapter 1. Most of this chapter will be review from 120; here are a few things to especially pay attention to:

- **Section 1.1.**
  You should know how to run Java programs from the command line, as well as from an IDE like Eclipse. For example
  `> java Cal 1927 3`
  runs the program in the file Cal.class with command line arguments "1927" and "3".
  `Cal.class` is created by compiling `Cal.java`, which must contain a `main()` method:
  ```
  public static void main(String [] args) { /* details of the method */ }
  ```
- **Section 1.2** `System.out.println()` and `System.out.print()`
- **Section 1.3** The eight primitive types in Java. Numeric, character, string and Boolean constants.
- **Section 1.4** Operators. Note that = (and the other = operators) return a value, so that z = x = Y+1; is allowed..
  Type conversion operators, such as `(double)` or `(int)`
- **Section 1.5** logical operators, short-circuit evaluation of && and ||, conditional statements and operator. A couple of things you might not have noticed before:
  short-circuit evaluation of && and || `break` and `continue` switch
  the conditional operator: **`?` `:`** It is like `if (..) .. else ..`, but it returns a value.

- `(3 < 2) ? 4 : 5` returns 5

- **Section 1.6** method header *vs.* method signature, formal parameters *vs.* actual arguments, overloading, static, final.
- **Exercises** 1.3, 1.5, 1.7, 1.8, 1.10, 1.13, 1.14, 1.18, 1.21

Chapter 2. Again, almost everything here should be review from CSSE 120 or 220. If any of these things are not review for you, or if you don't remember the details, read those parts of the chapter carefully.

- **2.1-1.2** Objects and references, new. The meaning of = and == when applied to objects.
- **2.3** Strings are immutable, concatenation, no comparison operators (use `equals` and `compareTo`), `length`, `charAt`, `substring`. Also look up the **StringBuffer** class in the JDK documentation.
- **2.4** Declaring an array of objects does not create an actual array object; use **new.** If the base types of the array is an object type, all of the array entries are initially null, unless that array is created by an **initializer**. = copies an array *reference*, not the array itself.
  Dynamic array expansion is the most important thing in this chapter. Figure 2.6
  - Notice the use of the InputStreamReader and BufferedReader wrappers for System.in. For now, think of lines 12-13 as a magic incantation for setting up line-by-line input.
  - Lines 21-24 are the standard way to read a text file line-by-line.
    - For an alternative, you may want to read about the **Scanner** class from JDK 1.5
  - Be sure to understand how *resize()* works (Figure 2.7).
  - Also why resize is called with argument array.length * 2

  Ragged 2D arrays
  Command line arguments
- **2.5** try, catch, finally, throw, throws. runtime exceptions, checked exceptions, errors.
- **2.6** I/O streams, **StringTokenizer**, opening files for reading and writing.
- **Exercises:** 2.1, 2.6, 2.7, 2.8, 2.10

Chapter 3. Even more of this Chapter should be review.

- **3.1-3.5** Objects, classes, methods, fields, `this`, static fields and methods, `instanceof`, static initializers.
- **3.6** Importing from packages, creating your own packages (search Java documentation or the internet to find out where to put files from packages if you do not use an IDE like JCreator).
- **3.7** The Composite (pair) pattern. Used for returning multiple values.
- **Exercises.**

Chapter 4. This chapter contains more things that you might not know if you did not pay very close attention in 220.

- **4.1-4.3** Inheritance basics: adding fields and methods, overloading *vs.* overriding, visibility (public, private, protected, package visible), type compatibility, dynamic binding and polymorphism, IS-A *vs.* HAS-A, use of `super` in constructors and methods, final methods and classes, the need for class cast, abstract classes and methods. Covariance.
- **4.4-4.5** Interfaces, the object class, decorator pattern and its application to IO classes.
- **4.6** Generic components, Wrapper classes, the adapter pattern, autoboxing/unboxing, interfaces for genericity
- **4.7** Java 1.5 generics
- **4.8** Function objects, Comparable vs. Comparator. Local, nested, and anonymous classes.
- **4.9** Dynamic binding applies to method selection, but we find that signature selection is static.
- **Exercises:** 4.6, 4.7-4.14, 4.18, 4.20-4.21, 4.27-4.29

Chapter 5. I list here the things I expect you to have gotten from 220. We will cover the rest in this course.

- **5.1** Basics of algorithm analysis, the huge difference in the growth rates of different kinds of functions, the fundamental questions on page 149. Why we look at big-Oh instead of exact running times.
- **5.2** More on classes of running times, and the kinds of algorithms that have these times.

- **5.6** (not 5.6.3) Sequential and binary search in a static array, analysis of these algorithms.
- **Exercises:** 5.3, 5.4, 5.6, 5.7, 5.8, 5.11, 5.12, 5.14, 5.15, 5.16, 5.17, 5.18, 5.20, 5.21, 5.22

Chapter 6. You should know the main data structures, interfaces, and classes of the Collections API.

- **Intro.** What is an abstract data type? A data structure? Specification, implementation, application.
- **6.1** The purpose of the weiss.nonstandard and weiss.util packages.
- **6.2** The iterator pattern, approaches to implementation. Factory methods.
- **6.3** Collections API approach to collections and iterators
- **6.4** Generic algorithms for arrays and collections (the **Arrays** and **Collections** classes). Comparators.
- **6.5** Lists, and their implementations as ArrayLists and LinkedLists.
- **6.6** Stacks, parenthesis matching, queues.
- **6.7-6.9** I will not assume that you know about these sections before this course begins.
- **Exercises:** 6.1, 6.2, 6.5, 6.8, 6.9, 6.10-6.15, 6.17, 6.18 (use the Arrays class).

Chapter 7. Recursion and induction

- **7.1** Basic ides of recursion.
- **7.3** Basic recursion, printing numbers recursively, Fibonacci numbers, four rules for recursion; recursive *definition* of "tree", recursive calculations of factorial, binary search.

Chapter 8 Sorting

- **8.1-8.3** Why is sorting important, interface to sorting algorithms. Insertion sort. Selection sort (not actually in this chapter, look it up on-line). Why is insertion sort slow? Why is bubble sort slower? What about selection sort? What is an inversion, and why does that matter.
- **8.5** Merge sort. You should be able to write merge and mergesort "from scratch" and to understand their analysis and limitations.
- **8.4 (not 8.4.1).** How is Shell Sort related to Insertion Sort? Why is it faster?
- **Exercises:** 8.7, 8.10, 8.15, 8.19, 8.27 (use the Arrays class).

CSSE 230 Spring term: A few more things

Links to the attachments referenced by this document appear at the end.

1. First, a reminder to get the software installed and get the textbook before you leave campus, as described in my last message.

2. Proofs by Mathematical induction have always been a significant part of CSSE 230, and if anything, the role will increase in the "new" 230. Since MA275 is no longer a pre-requisite for 230, for many of you, induction may be a brand new idea. It is an idea that some people find difficult. If abstract mathematical thinking does not come easy for you, you may want to get a head start by looking at some material on induction before the term begins. There is a brief introduction on pages 253-255 of the textbook. There are some additional good introductions, with several more simple examples, on the internet. I have attached a document that contains links to some of them.

3. I thought that some of the questions on this week's 220 final exam might help you to quickly discover whether there is a lot of 220 material that you need to review. There are three documents:

- The written problems from the exam. One of those problems (#5) may be difficult to do without the specific context of this term's course, You should be able to do the others.
- The computer problems from the exam
- A ZIP file of the Eclipse project that I gave the students as a starting point. In order to get the JUnit tests to work, you need to (after you have unzipped the project and imported it onto Eclipse),
    - Right-click on the project, and choose New … JUnit Test.
    - In the dialog that comes up, you'll probably see a message like "JUnit 3.8.1 is not on the build path of … Click here to add JUnit …"
    - Click on that "Click Here" link, then click Cancel.
    - Right-click one of the "…Tests.java" files, and choose Run As… JUnit Test.

Have a good break. I look forward to working with you next term. Be ready to "hit the ground running" on the first day of class.

Claude Anderson

Documents that were attached to the original email message:

MathematicalInductionLinks.pdf

FinalExamPaperPart.pdf

FinalExamCompPart.pdf

220ExamFinal-200820.zip